CVE-2016-9276

描述：The dwarf\_get\_aranges\_list function in dwarf\_arrange.c in Libdwarf before 20161124 allows remote attackers to cause a denial of service (out-of-bounds read).

软件：Libdwarf 20161021

源码： dwarf\_arrange.c

出现位置： 180=248行

关键源码：

do {

Dwarf\_Addr range\_address = 0;

Dwarf\_Unsigned segment\_selector = 0;

Dwarf\_Unsigned range\_length = 0;

/\* For segmented address spaces, the first field to

read is a segment selector (new in DWARF4).

Surprising since the segment\_size was always there

in the table header! \*/

if ((version >= 4) && (segment\_size != 0)) {

READ\_UNALIGNED\_CK(dbg, segment\_selector, Dwarf\_Unsigned,

arange\_ptr, segment\_size,

error,end\_this\_arange);

arange\_ptr += address\_size;

}

READ\_UNALIGNED\_CK(dbg, range\_address, Dwarf\_Addr,

arange\_ptr, address\_size,

error,end\_this\_arange);

arange\_ptr += address\_size;

READ\_UNALIGNED\_CK(dbg, range\_length, Dwarf\_Unsigned,

arange\_ptr, address\_size,

error,end\_this\_arange);

arange\_ptr += address\_size;

{

/\* We used to suppress all-zero entries, but

now we return all aranges entries so we show

the entire content. March 31, 2010. \*/

arange = (Dwarf\_Arange)

\_dwarf\_get\_alloc(dbg, DW\_DLA\_ARANGE, 1);

if (arange == NULL) {

\_dwarf\_error(dbg, error, DW\_DLE\_ALLOC\_FAIL);

return (DW\_DLV\_ERROR);

}

arange->ar\_segment\_selector = segment\_selector;

arange->ar\_segment\_selector\_size = segment\_size;

arange->ar\_address = range\_address;

arange->ar\_length = range\_length;

arange->ar\_info\_offset = info\_offset;

arange->ar\_dbg = dbg;

arange\_count++;

curr\_chain = (Dwarf\_Chain)

\_dwarf\_get\_alloc(dbg, DW\_DLA\_CHAIN, 1);

if (curr\_chain == NULL) {

\_dwarf\_error(dbg, error, DW\_DLE\_ALLOC\_FAIL);

return (DW\_DLV\_ERROR);

}

curr\_chain->ch\_item = arange;

if (head\_chain == NULL)

head\_chain = prev\_chain = curr\_chain;

else {

prev\_chain->ch\_next = curr\_chain;

prev\_chain = curr\_chain;

}

}

/\* The current set of ranges is terminated by

range\_address 0 and range\_length 0, but that

does not necessarily terminate the ranges for this CU!

There can be multiple sets in that DWARF

does not explicitly forbid multiple sets.

DWARF2,3,4 section 7.20

We stop short to avoid overrun of the end of the CU. \*/

} while (end\_this\_arange >= (arange\_ptr + range\_entry\_size));